Deakin, T., McIntosh-Smith, S., Lovegrove, J., Smedley-Stevenson, R., & Hagues, A. (2019). *Reviewing the Computational Performance of Deterministic SN Transport Sweeps on Many-Core Architectures*. Abstract from International Conference on Transport Theory, Paris, France.

The 26th International Conference on Transport Theory (ICTT-26)
Paris, France, September 23-27, 2019

REVIEWING THE COMPUTATIONAL PERFORMANCE OF DETERMINISTIC S_N TRANSPORT SWEEPS ON MANY-CORE ARCHITECTURES

Tom Deakin and Simon McIntosh-Smith
Department of Computer Science
University of Bristol
{tom.deakin, S.McIntosh-Smith}@bristol.ac.uk

Justin Lovegrove, Richard Smedley-Stevenson and Andrew Hagues
Atomic Weapons Establishment
Aldermaston, UK
{Justin.Lovegrove, Richard.Smedley-Stevenson, Andrew.Hagues}@awe.co.uk

In recent years the computer processors underpinning the large, distributed, workhorse computers used to solve the Boltzmann transport equation have become ever more parallel and diverse. Traditional CPU architectures have increased in core count, reduced in clock speed and gained a deep memory hierarchy. Multiple CPU vendors such as Intel, AMD and Marvell (Arm) are offering a collectively diverse range of processors. GPU accelerators are used to provide high levels of performance, with half of the top 10 supercomputers worldwide using GPU technology according to the Top 500 list [9]. Indeed, this list of the fastest machines highlights the growing diversity in many-core architectures, with the top 10 machines leveraging 8 different CPU architectures and 3 different GPU architectures between them. Going forward, the landscape of processor technology available will require all our codes to function well across multiple architectures. As such, this ever increasing range of architectures represents a unique challenge for solving the Boltzmann equation using deterministic methods in particular, and so it is important to characterise the performance of those key algorithms across the processor spectrum.

In this work, we will explore the performance profiles of a deterministic transport sweep occurring on both 3D structured (Cartesian) and unstructured (hexahedral) meshes. For structured grids, we will use the standard upwinded finite difference spatial discretisation. For unstructured meshes, we use a discontinuous Galerkin finite element method, following the 'matrix-free' methodology where the global matrix is not assembled, with only the local systems assembled and solved. Tensor products of 1D Lagrange polynomials of arbitrary order are used for our set of basis functions.

The spatial sweep across the mesh along a given angle in the S_N quadrature set forms the most expensive routine in terms of application runtime for such solvers. It is this key *kernel* that we explore, and examine its performance across a range of many-core architectures. In particular, we will make the following

contributions:

- We will survey the performance of transport sweeps across a very large selection of diverse architectures, including the latest CPUs and GPUs from a variety of vendors.
- The performance limiting factors for the key kernels will be discussed, and detail where future research is required in characterising performance.

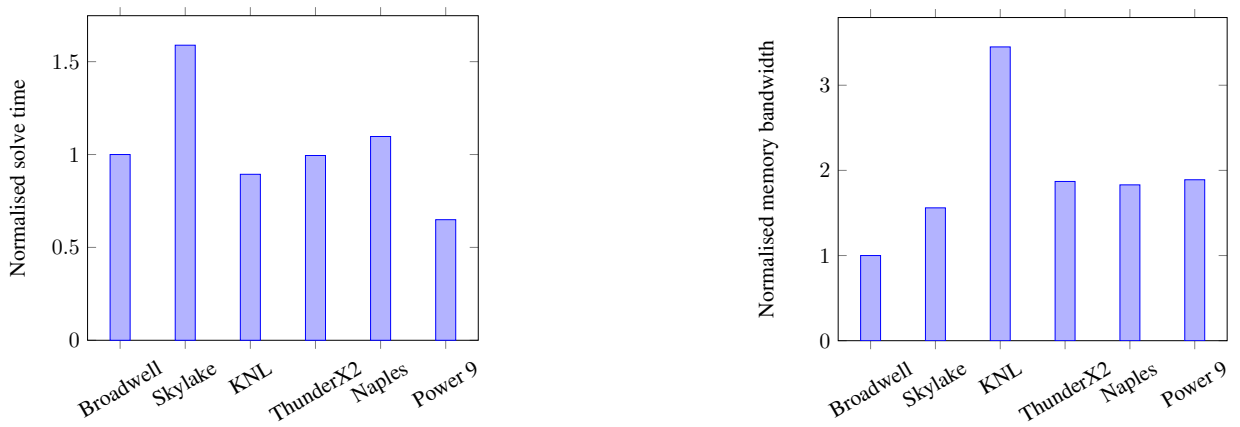
1 STRUCTURED SWEEPS

The **mega-sweep** application[†] is a mini-mini-application for **SNAP**, which is itself a mini-application for PARTISN (from Los Alamos National Laboratory) [3, 11]. These mini-apps seek to capture the performance profiles of transport sweep applications: SNAP of a modern transport code, and mega-sweep the performance characteristics of the sweep itself without the complexity of source iterations. The **KRIPKE** mini-application (from Lawrence Livermore National Laboratory) is an alternative proxy for structured sweeps which captures differing design decisions in writing a transport code [7]. Although ensuring timely convergence from a robust iteration scheme is important to the overall performance of a transport code, it is the sweep itself that occurs for each of these iterations that usually contributes to the majority of the runtime. In contrast, in KRIPKE the reduction of the angular flux into the scalar flux often contributes significantly to the runtime due to a lack of data reuse as a result of the choice of implementing the operators in the Boltzmann equation. Focussing on the sweep itself provides a more tractable approach for performance analysis. As such, mini-apps provide agile research vehicles where it is tractable to explore the fundamental properties of the algorithm without the burdens associated with production applications. Understanding the data movement of the transport kernels is a key focus of this work.

The mega-sweep code (and its parent SNAP) performs sweeps according to a KBA spatial decomposition using MPI, using the CPU vector units for updating all angles within each octant in parallel, and OpenMP threads over the energy groups under a Jacobi scheme (after [2]). Here, OpenMP worksharing directives are used on the energy group loop rather than the SPMD-style OpenMP programming used in SNAP; this SPMD approach requires high levels of thread support from the MPI library and is not compatible with GPU (offload) architectures. The octants are swept in sequential order, and the concurrency in this domain is not exposed so that the findings may be useful to a wider range of problems where parallel octant sweeps are not available (such as problems with reflective boundary conditions).

For this abstract, Fig. 1a shows a selected highlight of some of our performance results for mega-sweep across different CPU architectures, normalised to Broadwell. Under a traditional Roofline model, due to the low computational intensity of the sweep finite difference kernel, the performance would be classified as main memory bandwidth bound, and not bound by the rate of floating point operations (FLOP/s) [10].

[†] `https://github.com/UK-MAC/mega-stream`



(a) mega-sweep results

(b) BabelStream Triad results

Figure 1: mega-sweep and BabelStream Triad results on a range of CPU architectures, normalised to Broadwell (higher is better)

STREAM Triad is the classic benchmark for determining available main memory bandwidth, and codes which are main memory bandwidth bound typically align with the benchmark results [6, 8]. The results shown in Fig. 1b show that the sweep results of Fig. 1a do not correlate with being limited by main memory bandwidth. Indeed, processors with a high memory bandwidth do not necessarily provide fast runtimes for the sweep. This implies that neither FLOP/s or main memory bandwidth are a significant performance limiting factor for structured grid sweeps.

We will extend this analysis by exploring the use of hardware performance counters to identify bottlenecks in the code. We have used this approach in collaboration with Marvell to discover long latency instructions, and for where the hardware prefetcher is unable to correctly move memory into cache in advance of when it is needed. As such, the performance is limited primarily by loading memory from the cache hierarchy on multi-core CPUs. This is therefore an unusual performance characterisation as many other simulation algorithms are instead limited by main memory bandwidth instead, and is as such an important point to discuss within the transport community.

On GPU architectures on the other hand, additional concurrency in the algorithm must be exposed in order to obtain good performance. The natural independence between cells on the wavefront of the sweep provide this extra parallelism, and when combined with the concurrency in angles (within a single octant) and energy groups, sufficient parallelism is found to saturate a GPU with work [1, 4]. This extra parallelism leverages the latency hiding advantages of GPU architectures obtained by their ability to context switch quickly to hide long latency memory requests with other work. As a result, device memory bandwidth memory bandwidth becomes the performance limiting factor on GPU architectures.

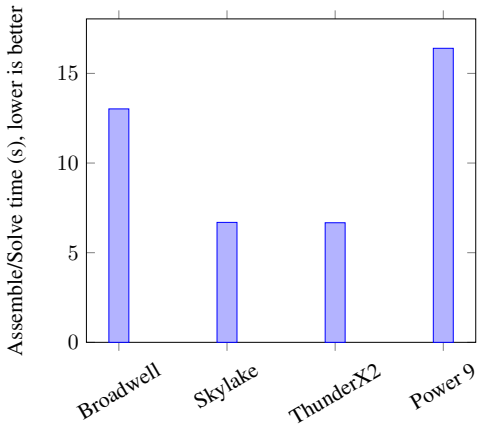


Figure 2: UnSNAP linear elements performance results

2 UNSTRUCTURED SWEEPS

Whilst solving the Boltzmann transport equation on a structured mesh requires solving simple diamond difference equations, solution on an unstructured grid requires a more involved method. We have developed the **UnSNAP** mini-app to explore the performance of using a 'matrix-free' discontinuous Galerkin finite element solution to the transport equation on unstructured hexahedral meshes [5]. The parallel scheme used vectorises over the nodes within each element, and uses OpenMP threads to parallelise over the energy group domain and cells within each wavefront. The performance results in Fig. 2 show the runtime on each CPU architecture for the main local element matrix assembly and solution, following the sweep ordering. It should be noted that the computational intensity of a structured grid finite difference discretisation and a linear finite element discretisation is similar, as although more floating point operations occur in the finite element method, more memory reads are required [3]. As found with the structured case, there is no correlation between the memory bandwidth of the architecture (Fig. 1b) and the performance of the transport solve. Therefore, a major performance limiting factor is again due to cache access as the small, dense local matrix is small enough to be cache resident and has high reuse. We will explore the cache behaviour further in our presentation.

3 SUMMARY

The sweep based algorithms forming a key component of neutron and thermal radiation transport codes will continue to be important on future architectures. It is important that the transport community explore the performance limiting factors of these algorithms to ensure a high level of performance is maintained on many-core processors. Although some multi-level approaches are being investigated as an alternative to sweeps, these face an equally daunting but significantly different set of challenges.

ACKNOWLEDGMENTS

The authors express their gratitude to their Institution.

REFERENCES

[1] David Appelhans, Steve Rennich, Adam Kunen, and Leopold Grinberg. GPU Optimization of the Kripke Neutron-Particle Transport Mini-App. In *GPU Technology Conference*, April 2016.

[2] Randal S Baker. An Sn Algorithm for Modern Architectures. In *Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method*, number ANS MC2015, Nashville, TN, 2015. American Nuclear Society.

[3] Tom Deakin. *Leveraging Many-Core Technology for Deterministic Neutral Particle Transport at Extreme Scale*. PhD thesis, University of Bristol, 2018.

[4] Tom Deakin, Simon McIntosh-Smith, and Wayne Gaudin. Many-Core Acceleration of a Discrete Ordinates Transport Mini-App at Extreme Scale. In M J Kunkel, P Balaji, and J Dongarra, editors, *High Performance Computing: 31st International Conference, ISC High Performance 2016, Frankfurt, Germany, Proceedings*, pages 429–448. Springer International Publishing, Cham, June 2016.

[5] Tom Deakin, Simon McIntosh-Smith, Justin Lovegrove, Richard Smedley-Stevenson, and Andrew Hagues. UnSNAP: A Mini-App for Exploring the Performance of Deterministic Discrete Ordinates Transport on Unstructured Meshes. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 598–606, Belfast, sep 2018. IEEE.

[6] Tom Deakin, James Price, Matt Martineau, and Simon McIntosh Smith. Evaluating attainable memory bandwidth of parallel programming models via BabelStream. *International Journal of Computational Science and Engineering*, 17(3):247–262, 2018.

[7] Adam J Kunen, Teresa S Bailey, and Peter N Brown. KRIPKE - A Massively Parallel Transport Mini-app. In *Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method*, number ANS MC2015, pages 1–13, Nashville, Tennessee, 2015. American Nuclear Society.

[8] John D McCalpin. Memory Bandwidth and Machine Balance in Current High Performance Computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pages 19–25, dec 1995.

[9] Erich Strohmaier, Horst Simon, Jack Dongarra, and Martin Meuer. Top 500 - November 2018. http://www.top500.org, 2018.

[10] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52:65–76, 2009.

[11] Robert J. Zerr and Randal S. Baker. SNAP: SN (Discrete Ordinates) Application Proxy - Proxy Description. Technical report, LA-UR-13-21070, Los Alamos National Labratory, 2013.